Algorithmic Trading with Bitcoin

CS 156, sec 01 - Artificial Intelligence

Alexis Candelaria
Emmanuel Mendoza
Stephen Piazza
Victor Fateh

Table of Contents

Introduction

The secret to living a prosperous life is by having multiple revenue streams. Investing is one of the best ways to increase your wealth through passive income. To be a good trader, you have to follow a strict system and not let your emotions decide for you. Fortunately computers provide the optimal tool for this kind of analysis and transaction based system.

Bitcoin is a digital open source currency that nobody has direct ownership of. Utilizing peer to peer connectivity and trading it is able to operate without any banks or central authority overseeing the transactions. This digital payment method was created in 2009 when the first client came online. Since then Bitcoin exchanges have been set up in a majority of developed nations with some international exchanges cropping up as well. These exchanges are similar to stock exchanges in which they are used as a platform to buy and sell bitcoins at various rates. An example of this format utilizing the European Bitcoin (EUR/BTC) can be seen in a buyer/seller transaction method. A seller will put a limit order for the desired price he wishes to sell his EUR/BTC, maybe 60 EUR. This can be higher or lower than the previous sell point. When a buyer comes along he will create a market order setting his criteria for buying at a certain price point. The system will take this information match buyers and sellers based on the criteria for buying being lower than the price of the Bitcoin. This creates upward and downward trends in the market that if examined correctly and invested at the proper time can be used to generate profit.

This project was spawned from the thought of maximizing profit using various trade strategies and machine learning. The hope that is when the system is fully implemented it would be able to successfully predict patterns occurring in the market that the trading strategy alone wouldn't be able to detect. In the end this will leave the user deploying this automated buying and trading agent with a short term strategy that will maximize the gain.

Background

There have been several studies done in machine learning involving bitcoin trading agents trying to utilize artificial intelligence to create maximum gain. A few of these studies were examined before implementation of the project began. A brief description and summary will be provided.

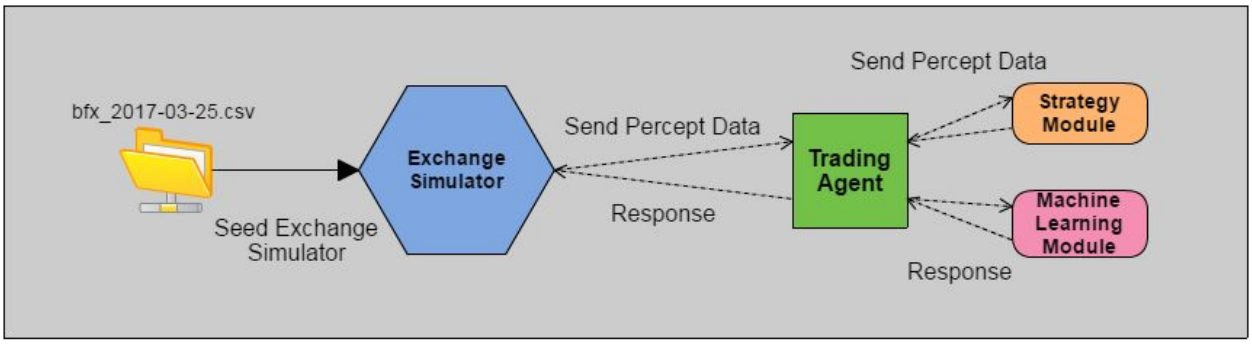
The first study provided was done by Tom Bell at the University of Southampton, School of Electronics and Computer Science. Bell first goes over the stability of Bitcoin as a currency. It is found that long term it has long term potential since the price volatility is correlated to the S&P 500 index. This shows that the market is being utilized by not only programmers and criminals as originally thought, but also being used by investors. This paper emphasizes the necessity to understand the underlying drivers of bitcoin prices to optimize your system. The agent used in this paper to attempt to predict future pricing is a Bayesian Regression for 'Latent Source Model' which uses previous prices in Bitcoin history training data to try and predict future prices. Using this method they claim to be able to make an 89% return over a 50 day period.

The second study was done by Devavrat Shah and Kang Zhang Of Massachusetts Institute of technology. This study was much more technical in nature describing many of the calculations that went into the prediction agent. In the end they also utilized Bayesian Regression as a predictive model of future prices. They indicate that they use a representative time series as the alternative would have been computationally too intensive, however it is feasible given the proper hardware. This would allow for more accurate predictions and an increase in gains created by the agent.

Application

The application takes in a stream of data from a bitcoin exchange called Bitfinex. Bitfinex provides a websocket api that provides exchange data for the price of bitcoin. Each tick of exchange data returns data about the current price of bitcoin, bid, bid size, ask, and ask size. The application continuously queries the bitfinex API and populates a csv from which the application will also evaluate trends given various trading strategies. Currently, the only trading strategy that has been implemented is the EMA Crossover strategy. What this means technically is that there are two moving windows one for the short period and one for the long period. These moving windows calculate the exponential moving average for both the long and short periods and looks for crossovers in these averages. The exponential moving average is a commonly used indicator for detecting the momentum of any stock. Calculating the exponential moving average as opposed to a simple moving average gives more weight to the latest values from the bitcoin exchange. Every 5 seconds the exchange is queried and the window is updated. Next, the updated data is then evaluated to see if an ema crossover has occurred and whether it is a buy signal or a sell signal. Then an optionally used machine learning component was also built that utilized naive bayes to build a classifier on whether or not the signal from the crossover strategy was correct. There was extensive use of pandas to conduct calculations over the populated csv. Pandas was coincidently created by an engineer in the finance industry, so it is well suited for algorithmic trading.

Agent Design

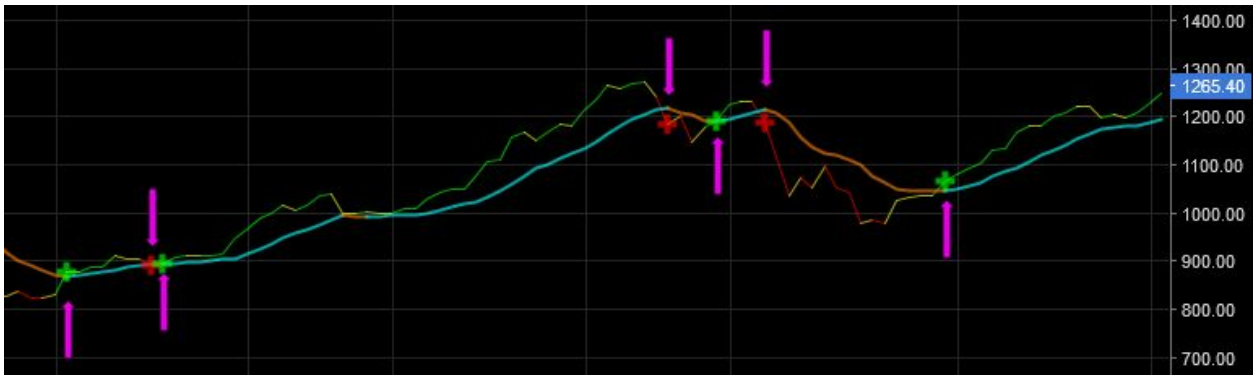


The agent's primary task is to pull its precepts from the csv as they were populated and to conduct its currently set trading strategy. The agent is designed in a way that alternative

trading strategies can easily be substituted for what is currently the EMA crossover strategy. The agent also optionally operates with a machine learning module that currently utilizes a naive bayes classifier to provide further insight on the success of the EMA strategy. The naive bayes classifier was chosen because it is simple to implement and it is fairly performant in comparison to other supervised learning models. The labels or classes that are utilized for this classifier are basically that the EMA crossover strategy was correct or incorrect. The Naive Bayes classifier makes a naive assumption that features are independent from one another. More specifically, a gaussian naive bayes classifier was utilized because it is particularly suited for continuous data. A gaussian naive bayes classifier makes a further assumption that the values for each feature of a class are distributed normally. The agent then calculates the mean and standard deviation for each feature of the tick data and produces a probability of being a part of each class. The class with the higher probability is the classification for that datapoint. The agent then runs this classifier and then waits until the next tick of data to come in to update the data point with a correct classification or incorrect classification. Since, the classifier has “learned” from a series of earlier ticks, the agent was able to experience a higher return on investment in an uptrend than when no naive bayes classifier was used. If the machine learning module is activated then the EMA crossover strategy will first consult the naive bayes classifier and the machine learning has the final say on whether an action is taken. This means that the strategy module could suggest a trade to the agent due to a crossover event but that no action can still be taken if the classifier classifies the signal as incorrect.

Agent Rule-Based Strategy

Graph of Bitcoin price movement with two different period EMA lines overlaid.



Graph of Bitcoin price movement with only the two different period EMA lines visible.

The exponential moving average (EMA) is an exponentially weighted moving average which produces more sensitive movements when compared to a simple moving average. Calculating an EMA helps visualize the price action of an investment by filtering out the random price fluctuations and producing a smoother line to model the price movement. Overlaying two EMA's with different period lengths can then be used to determine the overall trend of an investment by allowing you to track the rate of change for both EMA's. The shorter period EMA will produce a more sensitivity line movements than the longer period EMA, using this relationship can then be used to determine the trend of an investment. When the short period EMA crosses up through the long period EMA, this is an indication of a buy signal. When the short period EMA crosses down through the long period EMA, this is an indication of a sell signal. This is the base strategy that was used by our trading agent to trade bitcoin.

When the EMA strategy module evaluates that the price of bitcoin is entering an uptrend, it will return a buy signal to our trading agent which suggests to our agent to perform a market buy order for bitcoin using the agent's entire dollar balance. When EMA strategy module evaluates that the price of bitcoin is entering a downtrend, it will return a sell signal to our agent which suggests to our agent to perform a market sell order for dollars using its entire bitcoin balance.

Representation

The representation scheme that we used for this project was a Rule Based representation. This fit our project the best since we could catalog trades in our knowledge base and classify them as successful or not. Our knowledge base continued to increase with each trade that was executed When deciding to make a trade the machine learning algorithm would look at previous percepts and if successful in the given circumstance then it would make the trade. If the knowledge base was given a percept that showed it would not be successful then it would not initiate the market sell or market buy order.

Search Algorithm

The search algorithm that we utilized for this project was the hill climbing heuristic. The hill climbing search algorithm was utilized to find the optimal exponentiated moving average short period and exponentiated moving average long period for the price of bitcoin. To make the algorithm more efficient, we utilized historic data and ran the EMA crossover strategy across all different combinations of EMA with the short period ranging from 100 to 199 and the long period ranging from 101 to 200. The return on investment for each ema period combination was then saved in a csv file to expedite computation of the optimal return. Once the csv was populated, random ema short and long period lengths were chosen and then hill climbing search was initiated from this randomly chosen starting point. In most cases the EMA combination had 8 neighboring solutions. The 8 neighboring solutions were calculated by proving all the permutations of incrementing, decrementing, or leaving the original value of both short and long ema periods. For example if you have randomly chosen ema periods of 4 and 8 ticks then the

neighboring solutions would be 4:9, 4:7, 5:9, 5:7, 5:8, 3:9, 3:7, 3:9. Out of the 8 neighboring solutions, the max return on investment was evaluated and the EMA pair producing the highest return on investment is then recursively run through the hill climbing algorithm. The base case of the hill climbing search is reached when an EMA period combination's return on investment is higher than all of its neighbors. This is the case when a local optimum is found and thus returns at least a locally optimal EMA period pair. Finding a local optimum as opposed to an absolute maximum is an existing problem with the hill climbing search but it provided a simple heuristic for at least improving ROI on the utilized trading strategy.

Machine Learning

The machine learning algorithm that we chose for this project was Naive Bayes for its simplicity in understanding how it works. Naive Bayes assumes that the probability of each variable in each class in independent from every other variable. While this is a strong assumption, it turns out to be an effective technique when used with similar multivariable problems.

The way Naive Bayes was applied to our agent was to calculate the probability of a trade decision being a Win or Loss by feeding our machine learning module a set of perceived percept variables when a trade decision is made and calculating its conditional probability. By multiplying the conditional probabilities together for each percept variable, we generate the probability of that percept data belonging to the Win or Loss class. To make a prediction, we can calculate the probabilities of the currently perceived percept data belonging to the Win or Loss class and select the class value with the highest probability.

To construct the Win or Loss classes. We first train our agent with 50% of the data. When our strategy module returns a buy or sell signal. We feed all the perceived percepts at that frame moment and the signal type to our machine learning module which book-keeps those percepts until a future buy or sell signal is made. Once a future signal is made, we can then evaluate our new percepts with our previous percepts to see if our trade decision successfully bought low and sold high and vice versa. Once this comparison is done, we classify the previous percept data as a Win or Loss then append that data it to our Win or Loss data set then repeat the process with the currently perceived data.

Once training is complete. We will have two lists of several percept data points for both win or loss classes. We then summarize each class data set by calculating the mean and standard deviation of each variable and storing this summary for future predictions.

To make a prediction, our strategy module must first return a buy or sell signal. We then feed all the perceived percepts at that frame moment and the signal type to our machine learning module. From within our machine learning module, we use a Gaussian probability density function to estimate the probability of the given percept falling within our Win or Loss classes based on the summaries we generated from the training data sets. The machine learning algorithm then returns the Win or Loss class with the highest probability based on the percept data. This results in the machine learning module making the final decision of whether our trading agent should execute a trade.

Examples & Results

Search Algorithm

Whenever you run our search algorithm, a random starting point is used and a local maximum is found for that parameter region. Here are examples of three separate runs.

(175, 190)
::::::final value has been found:::::::
return %1.3
short period ema: 175
long period ema: 190

(153, 166)
::::::final value has been found:::::::
return %2.53
short period ema: 155
long period ema: 166

(168, 185)
::::::final value has been found:::::::
return %2.14
short period ema: 170
long period ema: 184

Agent Trading Results

All simulations were performed using a starting balance of 1 bitcoin. Final results are relative to the percentage difference based on the starting balance dollar value.

Uptrend Test Results using optimal EMA parameters found between 100-200

EMA Crossover Strategy Only	EMA Crossover Strategy + Machine Learning
EMA Short: 149 EMA Long: 184 Result %: 1.36 Start Balance: $ 907.31 Final Balance: $ 919.67	EMA Short: 149 EMA Long: 184 Result %: 2.3 Start Balance: $ 907.31 Final Balance: $ 928.15

Uptrend Test Results using optimal EMA+ML parameters found between 100-200

EMA Crossover Strategy Only	EMA Crossover Strategy + Machine Learning
EMA Short: 132 EMA Long: 196 Result %: 0.57 Start Balance: $ 907.31 Final Balance: $ 912.53	EMA Short: 132 EMA Long: 196 Result %: 3.52 Start Balance: $ 907.31 Final Balance: $ 939.2

Downtrend Test Results using optimal EMA parameters found between 100-200

EMA Crossover Strategy Only	EMA Crossover Strategy + Machine Learning
EMA Short: 192 EMA Long: 200 Result %: -5.94 Start Balance: $ 965.99 Final Balance: $ 908.65	EMA Short: 192 EMA Long: 200 Result %: -3.49 Start Balance: $ 965.99 Final Balance: $ 932.25

Downtrend Test Results using optimal EMA+ML parameters found between 100-200

EMA Crossover Strategy Only	EMA Crossover Strategy + Machine Learning
EMA Short: 130 EMA Long: 191 Result %: -6.15 Start Balance: $ 965.99 Final Balance: $ 906.55	EMA Short: 130 EMA Long: 191 Result %: -2.02 Start Balance: $ 965.99 Final Balance: $ 946.44

Future Work

This project has given us the experience into how machine learning can be integrated into algorithmic trading. While we only implemented a simplified model for our application, we did gain the foresight to see how we could improve our implementations in future works. This project showed us the possibilities of using different percepts, different strategies, different machine learning algorithms and different trading models.

For our agent, we only used the percepts provided by Bitfinex. This could be expanded upon by using additional custom percepts such as price acceleration, trading volume acceleration and even other more advanced percepts such as sentiment analysis on bitcoin related news.

For our agent, we only used the EMA crossover strategy, while this strategy is good for trending markets, there is a delay when being notified that a trend has changed. In future implementations, we could add additional trading strategies to improve the net gains of the agent, such as calculating the area difference of the double EMA's in the EMA crossover

strategy and use the tangent of the derivative of that area to signal a trade, this will solve the delay problem in the EMA crossover strategy but will also cause more false positives.

For our agent, we used naive bayes because of its simplicity in understanding how it works. However, we did discover other machine learning algorithms that could also be applied to algorithmic trading. Some of the other candidates included Linear SVC, K-Nearest Neighbor, Stochastic Gradient Descent and Neural Networks.

Lastly, we could also explore other trading models such as using machine learning to detect patterns in algorithmic trading and help identify other trading agents which could then be exploited to lose.

Description of Member Contribution and Work

Research Alexis Candelaria Emmanuel Mendoza Stephen Piazza Victor Fateh	
Data Collection Emmanuel Mendoza Alexis Candelaria	**Architecture Design** Emmanuel Mendoza Alexis Candelaria
Implementation Emmanuel Mendoza Alexis Candelaria Stephen Piazza Victor Fateh	**Report Writing** Emmanuel Mendoza Alexis Candelaria Stephen Piazza Victor Fateh

Appendix: How to Run the Application

Setup

To simplify setting up python library dependencies, we used Anaconda - Python 2.7 python interpreter, the same interpreter that was used for one of our in-class assignments.

Finding Local Optimal EMA Parameters

Run **search_algorithm.py**

Expected output

```
(153, 166)
::::::final value has been found:::::::
return %2.53
short period ema: 155
long period ema: 166
```

Each consecutive run will use a different parameter start location and will return different localized maximum parameters based within that region.

Running Agent Simulations

Run **agent_unitTesting.py**

Ema parameters are hard coded and can be changed @ ln 32-33
Exchange simulator input csv file is hard coded and can be changed @ ln 35

Expected output

```
()
('EMA Short:', 225)
('EMA Long:', 231)
('Result %:', 6.47)
('Start Balance: $', 907.31)
('Final Balance: $', 965.99)
```

Likewise, agent_unitTesting.py contains code to allow you to run brute force simulations within a given range to find the optimal parameters. Just uncomment @ ln 43 & 89 and set the ema parameter boundaries on @ ln 69 & 71

References

Bell, T. (n.d.). Bitcoin Trading Agents. Retrieved March 2, 2017, from http://www.tomjbell.co.uk//wp-content/uploads/2015/05/BitcoinTradingAgents.pdf

Brandvold, M., Molnár, P., Vagstad, K., & Valstad, O. C. (2015). Price discovery on Bitcoin exchanges. *Journal of International Financial Markets, Institutions and Money,36*, 18-35. doi:10.1016/j.intfin.2015.02.010

Devarat Shah and Kang Zhang, Bayesian regression and Bitcoin, CoRR arXiv:1410.1231, October 2014